REMARKS

The Examiner has rejected claims 1, 3-20, 22-38, and 40-45. Claims 2, 21, and 39 were previously canceled. Claims 1, 12, 13, 20, 31, and 38 have been amended to clarify the features of the invention. As a result, claims 1, 3-20, 22-38, and 40-45 are pending for examination with claims 1, 12, 13, 20, 31, and 38 being independent claims. The amendments made and the new claims added find support in the specification, and do not constitute new matter.

Applicants thank the Examiner for conducting the personal interview of May 7, 2005, with the Applicant's attorney. During that interview the Applicant's representatives discussed claims 1 and 13, and the Richter and Du references. Applicant's representatives discussed the limitations of the claims including "detecting" and "policy" with the Examiner. Applicant's Attorney thanks the examiner for removing the finality of the rejection.

The Examiner has rejected Claims 1, 12, 20 and 38 under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6330,569 to Baisley et al. ("Baisley") in view of U.S. Patent No. 6,412,109 to Ghosh et al. ("Ghosh"), further in view of U.S. Patent No. 5,832,271 to Devanbu ("Devanbu"), and further in view of U.S. Patent No. 6,308,163 to Du et al. "(Du)" . The Examiner states that it would have been obvious to one having ordinary skill in the art at the time the invention was made to "incorporate the teaching of Du to include policy for substitution into the system of Baisley, Ghosh and Devanbu". Applicants submit that the invention as claimed in Claim 1, 12, 20 and 38 is neither taught, described or suggested in Du, even in view of Baisley, Ghosh and Devanbu.

Nonetheless Applicants have amended Claim 1 to

call for: "creating an association between ones of the set of

original program segments and ones of the set of substitute program segments, each association comprising a program matching criteria for determining whether to execute the substitute program segment depending on an identity of a calling module of the original program segment." (underlining added for emphasis)

Nonetheless Applicants have amended Claim 12 to call for:

"modifying executable binary code of the copy of the executable program in active process space to include an exception inducing code, wherein the exception inducing code comprises executable binary code that induces execution of an exception handling routine for determining and executing at least a corresponding one of the set of modifications; and associating, with each of the set of modifications, a program matching criteria for determining whether to execute the modified executable binary code depending on an identity of a calling module of the modified executable binary code." (underlining added for emphasis)

Nonetheless Applicants have amended Claim 20 to call for:

"creating an association between ones of the set of original program segments and ones of the set of substitute program segments, each association comprising a program matching criteria for determining whether to

execute the substitute program segment depending on an identity of a calling module of the original program segment." (underlining added for emphasis)

Nonetheless Applicants have amended Claim 38 to call for:

"a substitution list including entries describing an association between ones of the set of original program segments and ones of the set of substitute program segments, each association comprising a program matching criteria for determining whether to execute the substitute program segment depending on an identity of a calling module of the original program segment." (underlining added for emphasis)

The present invention provides:

"Modification specifications 200, described in greater detail below with reference to Fig. 4, include a set of executable program entries corresponding to programs for which modifications are specified in the database. Each executable program entry includes a program matching criteria and one or more references to substitute program segments. The program matching criteria comprises, by way of example, a set of designated tagged/typed values for a particular executable. In an embodiment of the invention, each program matching criteria is created from an extensible set of potential types

of information supported by the program modification database 192. A particular matching criteria for an executable entry is preferably a combination of values of selected types that, in combination, uniquely identify a program." (application page 13, lines 17-26)

Du, on the other hand provides for

"The method and the system provide a capability-based resource hierarchy for modeling static resource characteristics and a rule-based resource policy manager for dealing with dynamic resource properties.

In order to accommodate wide distribution of enterprise workflow resources across organizational and physical boundaries, resource management is also distributed. To support different views of enterprise workflow resources, global resource managers (GRMs) are subdivided into Enterprise GRMs (ERMs) and Site GRMs (SRMs). ERMs represent the enterprise-wide view of workflow resources and interface with the underlying SRMs, which represent partial views of workflow resources based upon organizational or physical boundaries.

There may be more than one level of SRMs in order to represent different levels of views. Thus, a tree hierarchy of resource managers is formed, with ERMs as roots." (Du column 4, lines 49-57)

The element does not have all of the limitations found in the collection of references. Du describes a basic "a rule-based resource policy manager for dealing with dynamic resource properties". Du does not describe a "program matching criteria and one or more references to substitute program segments". Applicants submit that the claimed "program matching criteria" has nothing to do with "workflow processing of an enterprise" as asserted by the Examiner. Du does not describe a "program matching criteria", and, as such, the Applicant submits that it has so relation to the present invention.

Accordingly, the Applicants submit that Claim 1 is not unpatentable over Du, even in view of Baisley, Ghosh and Devanbu. Claims 3-11 are dependent on Claim 1. As such, Claims 3-11 are believed allowable based upon Claim 1.

Accordingly, the Applicants submit that Claim 12 is not unpatentable over Du, even in view of Baisley, Ghosh and Devanbu.

Accordingly, the Applicants submit that Claim 20 is not unpatentable over Du, even in view of Baisley, Ghosh and Devanbu. Claims 22-30 are dependent on Claim 20. As such, Claims 22-30 are believed allowable based upon Claim 20.

Accordingly, the Applicants submit that Claim 38 is not unpatentable over Du, even in view of Baisley and Ghosh.

Claims 40-45 are dependent on Claim 38. As such, Claims 40-45 are believed allowable based upon Claim 38.

The Examiner has rejected Claims 13 and 31 under 35 U.S.C. §103(a) as being unpatentable over Richter ("Richter") in view of U.S. Patent No. 6,412,109 to Ghosh ("Ghosh"), and U.S. Patent No. 6,308,163 to Du et al. ("Du"). The Examiner states that it would have been obvious to one having ordinary skill in the art at the time the invention was made to "incorporate the teaching of Du to include policy for substitution into the system of Richter and Ghosh". Applicants submit that the invention as claimed in Claims 13 and 31 is neither taught, described or suggested in Richter, even in view of Ghosh and Du.

Nonetheless Applicants have amended Claim 13 and 31 to call for:

> "calling, from a calling module having an identity, an original program segment of the active executable program that has been modified to include a previously placed exception inducing code, the previously placed exception inducing code comprising executable binary code that induces an exception handled by an exception handling routine" and "detecting the previously placed exception inducing code." (underlining added for emphasis)

The present invention provides for:

> "An exclusion/inclusion policy column 326 stores conditions under which API substitution should occur or not occur. In an embodiment of the invention, the contents of an entry within the policy column 326 comprise an ordered list of conditions (e.g., routines or

modules that call the API containing the CLI), and an associated direction to include or exclude substitution when the condition is met. Due to the ordering of the list by precedence, the first list entry for which a condition is met controls whether to substitute for the originally called API. In an embodiment of the invention, if no conditions are present in an entry within policy column 326, then substitution for the originally called API is unconditionally performed. (application p 24; lines 11-19)

.....

Before describing step 408, the contents of modules that specify replacement functions for existing APIs are briefly described. In the disclosed exemplary embodiment of the invention, API hook descriptions are collectively stored within modules containing the substitute functions (e.g., APIs). In the exemplary embodiment, the modules are DLLs (referred to herein as "hook DLLs"). The "MyShim.dll" referenced by the <DLL> tag 270 in Fig. 4 is an example of such a module. Within the hook DLLs, API hook descriptions include: (1) the name of a module (e.g., DLL) containing an original function (e.g., API) that needs to be modified in some manner, (2) the name of the original function, and (3) a substitute function identification (e.g., name). In an alternative embodiment of the invention, the API hook descriptions and replacement functions are individually identified

within the database rather than grouping them within a hook DLL.

During step 408, the shim core/program loader 193 queries the database 192 to determine whether API hooks corresponding to the loaded executable program exist. If during step 408 the shim core/program loader 193 determines the database 192 specifies API hooks (e.g., one or more hook DLLs) for the loaded program, then control passes from step 408 to step 410. "
(Application page 25, line 25 to page 26 line 10)
(underlining added for emphasis)

Richter, on the other hand provides for:

"API Hooking by Overwriting Code
API hooking isn't new-developers have been using APT hooking methods for years. When it comes to solving the problem I just described, the first "solution" that everyone comes to is to hook by overwriting code. Here's how this works:

1. You locate the address of the function you want to hook in memory (say ExitProccss in Kcrncl32 .dll).
2. You save the first few bytes of this function in some memory of your, own.
3. You overwrite the first few bytes of this function with a JUMP CPU instruction that jumps to the memory address of your replacement function. Of course, your replacement function must have exactly the same signature as the function you're hooking:

all the parameters must be the same, the return value must be the same, and the calling convention must be the same.

4. Now, when a thread calls the hooked function, the JUMP instruction will actually jump to your replacement function. At this point, you can execute whatever code you'd like.

5. You unhook the function by taking the saved bytes (from step 2 and placing them back at the beginning of the hooked function.

6. You call the hooked function (which is no longer hooked), and the function performs its normal processing.

7. When the original function returns, you execute steps 2 and .3 again so that your replacement function will be called in the future."

(Richter p796, API Hooking by overwriting code, steps 1-7)

Richter does not disclose . "detecting the previously placed exception inducing code", and, as such, the Applicant submits that it has so relation to the present invention.

Accordingly, the Applicants submit that Claim 13 is not unpatentable over Richter in view of Ghosh and Du. Claims 14-19 are dependent on Claim 13. As such, Claims14-19 are believed allowable based upon Claim 13.

Accordingly, the Applicants submit that Claim 31 is not unpatentable over Richter in view of Ghosh and Du. Claims 32-37 are dependent on Claim 31. As such, Claims 32-37 are believed allowable based upon Claim 13.

CONCLUSION

Accordingly, in view of the above amendment and remarks it is submitted that the claims are patentably distinct over the prior art and that all the rejections to the claims have been overcome. Reconsideration and reexamination of the above Application is requested. Based on the foregoing, Applicants respectfully requests that the pending claims be allowed, and that a timely Notice of Allowance be issued in this case. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicants hereby request any necessary extension of time. If there is a fee occasioned by this response, including an extension fee that is not covered by an enclosed check please charge any deficiency to Deposit Account No. 50-0463.

Respectfully submitted,

Microsoft Corporation

Date: 6/2/05

By: Paul Heynssens

Paul B. Heynssens, Reg. No.: 47,648
Attorney for Applicants
Direct telephone (425) 707-3913
Microsoft Corporation
One Microsoft Way
Redmond WA 98052-6399

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

☒ deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

☐ transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

06-03-05
Date

Sherry Smith
Signature

Sherry Smith
Type or Print Name

Type of Response: Amendment
Application Number: 09/801,951
Attorney Docket Number: 154596.01
Filing Date: 3/8/01